Tema 3: Träd

Mattin Lotfi `malo5163`

17 februari 2016

1 Frågor

1. Får det finnas noder som har samma värde i ett binärt träd?

2. Vad är det positiva samt negativa med ett AVL-träd?

2 AVL-Träd

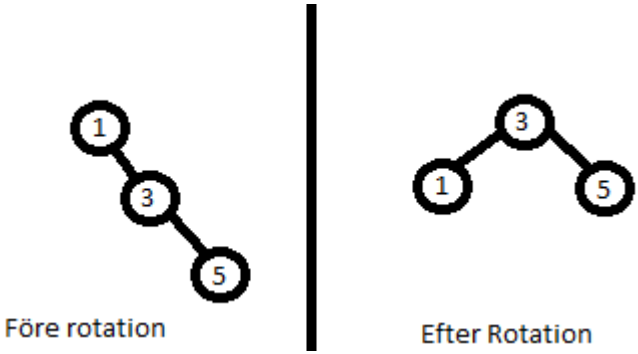
Introduktion

Ett AVL-träd är ett binärt sökträd som är självbalanserande. Det problem som uppstår med binära sökträd är att de ofta blir obalanserade, det innebär också att det ibland kan ske O(n) när trädet sedan skall läsas. Detta inträffar lätt då den lista som är i det binära sökträdet är obalanserad, t.ex. en gren som går jättelångt åt vänster och därmed blir vänstertung. Ett AVL-träd kan nästan alltid hantera trädet i O(Log N) då det kommer med

hjälp av rotationer balansera ut trädet. När en ny nod läggs till i trädet och gör trädet obalanserad kommer rotationerna ske och därmed balansera ut trädet igen.

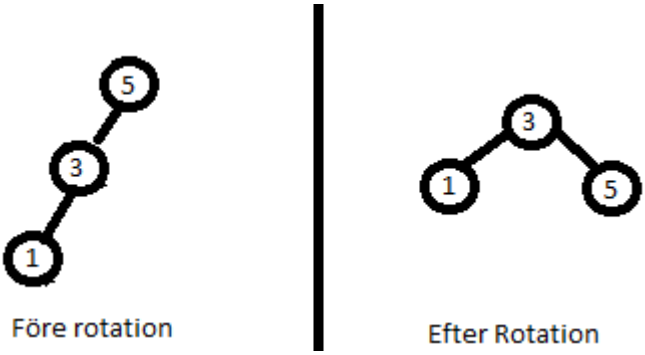
AVL-trädet består som tidigare nämnt av rotationer som sker och därmed roterar de noder som inte är i balans. Detta sker genom att en balanseringsfaktor som gör så att noderna skall ha 1, 0 eller -1. För att fastställa balanseringsfaktor så subtraheras den högra-grenens djup med den vänstra-grenens djup. Med denna formel så visar det att om svaret blir negativt är trädet högertungt annars om svaret är positivt så visar det trädet är vänstertungt. Genom olika operationer som kallas rotationer så balanseras trädet upp tills den uppnår balanseringsfaktorn, exempel finns nedan. Ett AVL-träd använder sig av samma logik som ett binärt sökträd när det gäller positioneringen av noder dvs mindre än åt vänster och tvärtom för att noden skall hamna åt höger. [1, sid. 123]

Vänster rotation



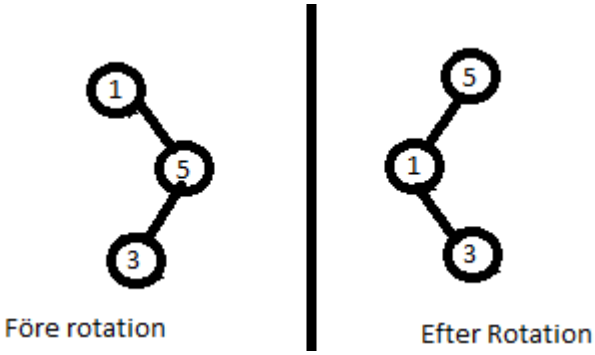
Den senaste noden som har lagts till i exempelt är 5.På den vänstra bilden ser vi att trädet är högertungt, då balanseringsfaktorn inte är uppfylld då vi i detta fallet subtraherar 0 med 2, vi får fram det negativa summan -2 och därmed måste vi balansera trädet. Då summan blev -2 så måste en vänster rotation ske. Det som sker är att 3 i detta fallet blir root och tar 1 som sitt vänstra barn och behåller 5 som sitt högra barn. När vi nu testar balanseringsfaktorn igen så kör vi formelen och får fram 1-1=0 och fyller därmed balanseringsfaktorn. [1, sid. 125-128]

Höger rotation

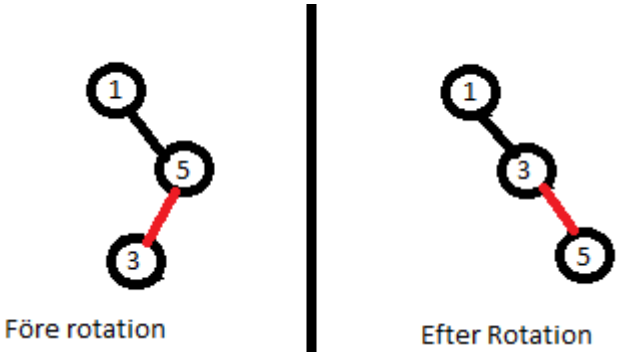


Den senaste noden som lagts till är 1. Höger rotation sker på samma sätt som vänster rotationen men spegelvänt. balanseringsfaktor är inte uppfyllt på den vänstrabilden då summan av balanseringsfaktorn blir 2, summan är nu positiv och därmed skall en högerrotation utföras. Det som händer i denna bild är att ”tre” blir root och tar ”fem” som sitt högra barn. [1, sid. 125-128]

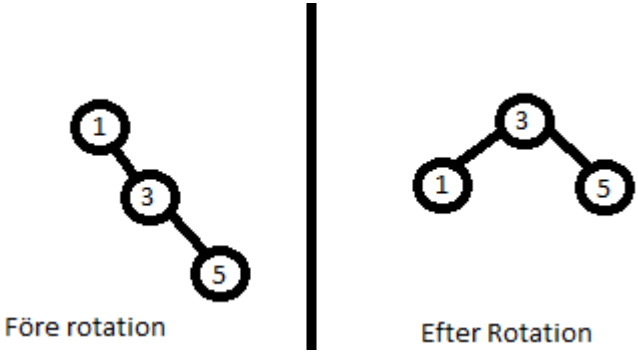
Vänster-Höger rotation



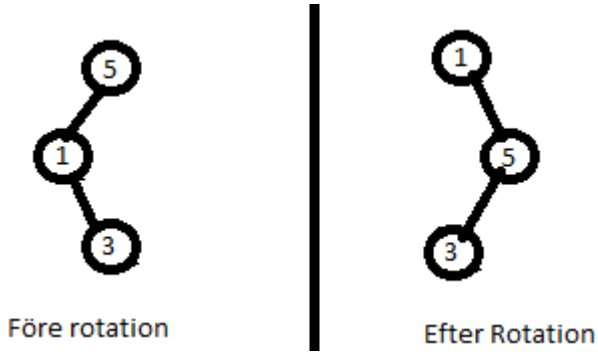
På bilden ser vi att alla noder är på höger sida och väljer då att göra en vänster rotation igen, men resultatet som sker är inte balanserat. Vi måste därför tänka om hur rotationen skall gå till för att få det balanserat. Vi skall därför göra en rotation i subträdet, det vill säga att vi inte kommer röra roten ännu utan bara ”5” och ”3”. På subträdet kommer vi göra en högerrotation för att barn noden är på vänster led.



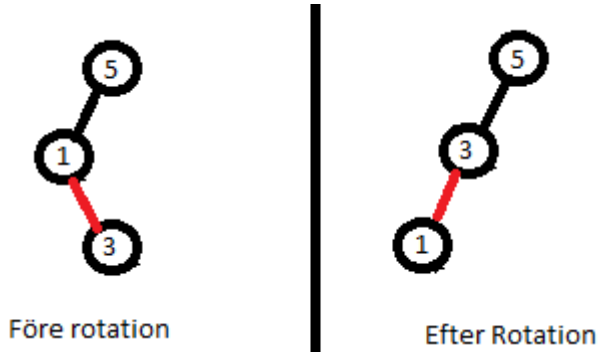
Vi ser nu att efter rotationen av subträdet har skett så ser den ut som före bilden i vänster rotations exempel, därmed vet vi nu att vi kan göra en vänster rotation för att balansera ut trädet ordentligt. [1, sid. 128-135]



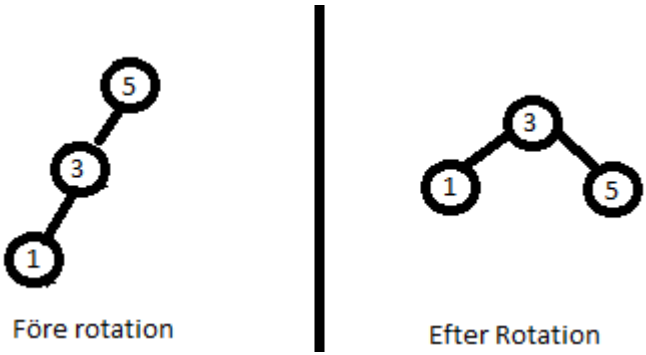
Höger-Vänster rotation



Vi ser att resultatet av att bara göra en höger rotation blir fel i detta fallet också, om vi skulle göra en vänster rotation nu skulle resultatet bli det samma som i före exemplet. Vi skall nu rotera subträdet med vänster rotation för att sedan göra en höger rotation



Vi har nu fått fram samma efter rotation som vi har i bild exemplet på höger rotation och kommer därmed att använda en höger rotation för att balansera trädet. [1, sid. 128-135]



Summering

I de exempel jag har tagit upp så har jag använt mig av väldigt små träd. Även om trädet är större så följer AVL-träd samma princip oberoende av storleken.

Referenser

[1] Mark Allen Weiss. *Data Structures and Algorithm Analysis in Java.* Pearson Education, third edition, 2012.